AMENDMENTS TO THE CLAIMS

1. (currently amended) A computer system for transferring data between a receiving central processing unit (CPU) and a transmitting CPU by using only write operations therebetween for the purpose of avoiding a direct read operation by the transmitting CPU from the receiving CPU, said system comprising:

a) at least one receiving central processing unit (CPU) comprising at least a read head register, a first queue length register, and a first total read register;

b) at least one transmitting CPU comprising at least a write head register, a second total read register, a total write register, and a second queue length register;

c) a local memory for said receiving CPU;

d) a local memory for said transmitting CPU;

e) means for connecting between said receiving CPU and said transmitting CPU where such means transfers write operations faster than read operations; and

f) a circular queue defined between designated addresses in said local memory of said receiving CPU, wherein said read head register contains a pointer to the location of the next read from said circular queue and said write head register contains a pointer to the location of the next write into said circular queue~~;~~, and

~~g) means for periodically updating said second total read register with the content of said first total read register; and~~

~~h) means for adding and updating at least a message separator between messages, such that said transmitting~~

~~CPU performs a read operation from said receiving CPU by the performance of:~~

~~1) a write operation providing a separator to said local memory of said receiving CPU at a location pointed to by said write head register, and at least one message to said local memory of said receiving CPU at a location pointed to by said write head register, requesting data to be read by said transmitting CPU, and~~

~~2) a write operation performed by said receiving CPU to said transmitting CPU containing said data to be read by said transmitting CPU~~

wherein said receiving CPU is coupled to periodically update said second total read register in the transmitting CPU with a content of said first total read register using one or more write operations, and

wherein said transmitting CPU is coupled to write data comprising one or more message separators and one or more messages into said circular queue, responsively to said second total read register, said total write register and said second queue length register.

2-3. (canceled)

4. (previously presented) The system of claim 1 wherein said means for connecting between said receiving CPU and said transmitting CPU is a PCI bus.

5-9. (canceled)

10. (previously presented) The system of claim 1 wherein said read head register and said write head register are set to point to the same address upon initialization.

11. (previously presented) The system of claim 1 wherein a maximum length is imposed on a message to be written into said circular queue.

12. (previously presented) The system of claim 11 wherein a tail is added at the end of said circular queue, said tail being equal in length to said maximum length imposed on said message.

13. (previously presented) The system of claim 1 wherein a header separator is used to indicate the end of said message.

14. (previously presented) The system of claim 13 wherein said header separator contains the length of the immediately following message.

15. (previously presented) The system of claim 13 wherein said header separator contains a predefined header "magic" number.

16. (previously presented) The system of claim 15 wherein, if said header separator contains an erroneous header "magic" number, an error message is generated.

17. (previously presented) The system of claim 13 wherein a separator of the last message in said queue is a stopper separator, and is different from said header separator placed between subsequent messages.

18. (previously presented) The system of claim 17 wherein said stopper separator further contains a predefined stopper "magic" number.

19. (previously presented) The system of claim 18 wherein, if a stopper separator contains an erroneous stopper "magic" number, an error message is generated.

20. (withdrawn) A method for writing a data message into a receiving queue between memory segments separated by a data bus comprising the steps of:

a) checking that the length of said message is not greater than the length of said queue, and generating an error message if said message length is greater than said queue length;

b) checking that length of said message is not greater than a maximum message length, and generating an error message if said message length is greater than said maximum message length;

c) repeatedly checking if there is sufficient memory available in said queue to contain said message until such time that sufficient memory is available;

d) writing said message into said queue;

e) writing a stopper designator immediately after the end of said message;

f) replacing the stopper designator at the end of the immediately preceding message with a message separator; and

g) updating the content of a total writes register by adding the number of bytes written into said queue.

21. (withdrawn) The method of claim 20 where said availability of memory in said queue is checked by a transmitting CPU by comparing said message length with the difference between said queue length and the difference between the total data written and total data read.

22. (withdrawn) The method of claim 20 further comprising aligning said message to a predefined alignment scheme by adding alignment padding bytes at the end of said message.

23. (withdrawn) The method of claim 22 where said alignment is on a four byte granularity.

24. (withdrawn) The method of claim 22 further comprising assigning the new address of the write head by calculating the sum of the old write head plus said

message length plus any applicable padding, plus the lengths of said beginning message separator and said stopper designator.

25. (withdrawn) The method of claim 22 where, in the event that said data message was first written into the tail of said queue, calculating said new address of write head further comprises deducting said the queue length.

26. (withdrawn) A method for reading a data message from a transmitting CPU into a queue with a tail of a receiving CPU, comprising:

a) checking that said message to be read is in said queue or otherwise wait for said message to enter said queue;

b) checking that the "magic" number is valid, or otherwise generate an error message; and

c) reading said message without alignment padding bytes.

27. (withdrawn) The method of claim 26, further comprising calculating the number of said alignment padding bytes added to said message.

28. (withdrawn) The method of claim 27, further comprising calculating the new address of the read head as the sum of the old read head plus the length of the message, plus alignment padding, plus the lengths of the stoppers, in the event that said data read is not from said tail of said queue.

29. (withdrawn) The method of claim 28 wherein said calculating further comprises deducting said length of said the queue, when said data was read from said tail of said queue.

30. (withdrawn) The method of claim 27 wherein said number of alignment padding bytes is added to the contents of the total read register.

31. (withdrawn) The method of claim 30 wherein the content of the total read register is written into the corresponding register of the memory of said receiving CPU.

32. (withdrawn) The system of claim 1, wherein writing a data message into said circular queue comprises the steps of:

a) checking that the length of said data message is not greater than the length of said queue, and generating an error message if said data message length is greater than said circular queue length;

b) checking that length of said data message is not greater than a maximum message length, and generating an error message if said message length is greater than said maximum message length;

c) repeatedly checking if there is sufficient memory available in said queue to contain said message until such time that sufficient memory is available;

d) writing said data message into said queue;

e) writing a stopper designator immediately after the end of said message;

f) replacing the stopper designator at the end of the immediately preceding message with a message separator; and

g) updating the content of a total writes register by adding the number of bytes written into said circular queue.

33. (withdrawn) The system of claim 12, wherein reading a data message from a transmitting CPU into said queue having said tail, comprises the steps of:

a) checking that said data message to be read is in said queue or otherwise wait for said message to enter said queue;

b) checking that the "magic" number is valid, or otherwise generate an error message; and

c) reading said message without alignment padding bytes.

34. (new) The system of claim 1, wherein said transmitting CPU is coupled to calculate a difference between said total write register and said second total read register, to subtract the difference from said second queue length register to produce a calculated available space, and to compare said calculated available space to the length of the one or more messages, so as to verify that sufficient memory space is available in the circular queue for writing said one or more messages.